

# Realización de Pruebas de Penetración en OWASP Juice Shop

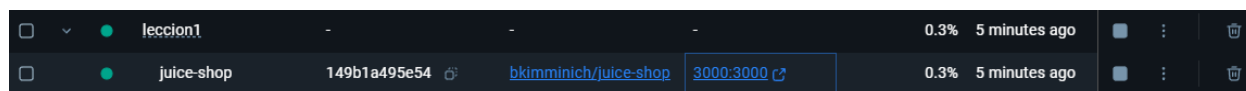
**Objetivo del ejercicio:** Realizar pruebas de penetración avanzadas en la aplicación web vulnerable OWASP Juice Shop para identificar y explotar vulnerabilidades de seguridad en diferentes capas de la aplicación. Se utilizaron herramientas como OWASP ZAP o Burp Suite para automatizar y realizar pruebas manuales, además de aprender a mitigar estas vulnerabilidades.

## 1. Escenario

El ejercicio se llevó a cabo en el entorno vulnerable de OWASP Juice Shop, diseñado para enseñar sobre las mejores prácticas de seguridad en aplicaciones web. El proceso de pentesting se basó en la metodología sugerida, que incluye las fases de exploración, escaneo automatizado, explotación y mitigación.

## 2. Acceso a la Aplicación

La aplicación web vulnerable, OWASP Juice Shop, se inició correctamente en un entorno local utilizando Docker. Esto permitió tener un entorno de prueba controlado para la realización de los ataques.



|                          |   |            |              |   |                                       |           |      |               |   |   |   |
|--------------------------|---|------------|--------------|---|---------------------------------------|-----------|------|---------------|---|---|---|
| <input type="checkbox"/> | ▼ | ●          | leccion1     | - | -                                     | -         | 0.3% | 5 minutes ago | ■ | ⋮ | 🗑 |
| <input type="checkbox"/> | ● | juice-shop | 149b1a495e54 | 🔗 | <a href="#">bkimminich/juice-shop</a> | 3000:3000 | 0.3% | 5 minutes ago | ■ | ⋮ | 🗑 |

## 3. Exploración y Pruebas con Herramientas de Seguridad

Se realizó una exploración manual de la aplicación, navegando a través de las páginas de registro e inicio de sesión y los campos de búsqueda. Se utilizaron herramientas como Burp Suite y las herramientas de desarrollo del navegador para interceptar y revisar las solicitudes de red.

### Hallazgos Clave:

Durante la exploración y el escaneo con herramientas, se identificaron y explotaron al menos tres vulnerabilidades críticas:

#### A. Vulnerabilidad de Inyección SQL (SQL Injection)

- **Descripción:** Se detectó que el campo de "Dirección de correo" en la página de inicio de sesión es vulnerable a la inyección SQL. Al ingresar un payload malicioso, es posible evadir la autenticación.

- **Explotación:** Se utilizó el payload ' ' OR 1=1-- ' en el campo de correo, lo que permitió eludir el proceso de inicio de sesión. El símbolo

-- al final comenta el resto de la consulta SQL, logrando que la condición 1=1 siempre sea verdadera.

- **Impacto:** Permite a un atacante no autenticado obtener acceso a la cuenta de cualquier usuario, incluyendo cuentas administrativas, comprometiendo la confidencialidad y la integridad de los datos.

## B. Vulnerabilidad de Cross-Site Scripting (XSS)

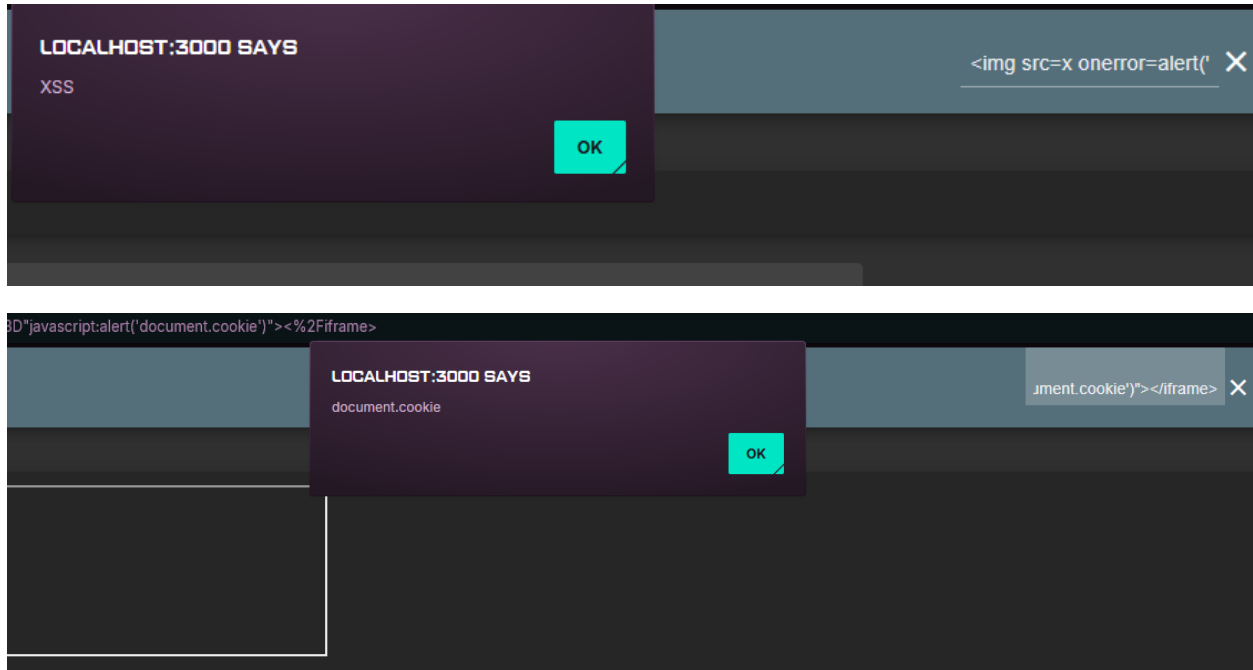
- **Descripción:** Se encontró una vulnerabilidad de XSS en el campo de búsqueda, lo que permite la ejecución de código JavaScript en el navegador del usuario. A pesar de que los payloads comunes con la etiqueta

<script> fueron filtrados, se logró evadir esta protección.

- **Explotación:** La vulnerabilidad se explotó utilizando payloads que se ejecutan a través de eventos en etiquetas HTML permitidas, como la etiqueta <img>.
  - Se inyectó el payload <img src=x onerror=alert('XSS')> para demostrar la ejecución de código, lo que resultó en una ventana de alerta.
  - Se demostró cómo robar información sensible, como las cookies de sesión, inyectando un script que las muestra en una alerta (

<iframe src="javascript:alert('document.cookie')"></iframe>).

- **Impacto:** Un atacante puede ejecutar scripts maliciosos en el navegador de un usuario, lo que podría llevar al robo de cookies de sesión, credenciales, o la manipulación de contenido en la página.



### C. Exposición de Datos Sensibles (CSRF & Otros)

- **Descripción:** Aunque no se explotó un ataque CSRF completo en este informe, se identificaron debilidades que lo harían posible. Además, se observó que la aplicación expone datos sensibles a través de solicitudes de red.
- **Explotación:**
  - **Inspección de Tráfico:** Utilizando Burp Suite y las herramientas de desarrollo de Google Chrome, se interceptaron y analizaron las solicitudes HTTP.
  - **Exposición de Datos:** En las respuestas de la API (/rest/user/whoami), se pudo observar información del usuario como el ID y el correo electrónico, lo que podría ser aprovechado por un atacante.
  - **Token de Autenticación:** Se identificó que la aplicación utiliza un token de autenticación (JWT) almacenado localmente para mantener la sesión, lo cual puede ser robado a través de ataques de XSS.

Burp Project Intruder Repeater View Help
 Burp Suite Community Edition v2025.5.5 - Temporary Pro

Dashboard Target **Proxy** Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer

Intercept HTTP history WebSockets history Match and replace Proxy settings

Intercept on Forward Drop

| Time             | Type | Direction | Method | URL  |
|------------------|------|-----------|--------|--|
| 18:56:43 26 J... | HTTP | → Request | GET    | http://localhost/  |
| 18:57:28 26 J... | HTTP | → Request | GET    | http://localhost:3000/assets/i18n/en.json                          |
| 18:57:28 26 J... | HTTP | → Request | GET    | http://localhost:3000/socket.io/?EIO=4&transport=polling&t=PUkTTYI |
| 18:57:28 26 J... | HTTP | → Request | GET    | http://localhost:3000/rest/admin/application-version               |
| 18:57:28 26 J... | HTTP | → Request | GET    | http://localhost:3000/api/Challenges/?name=Score%20Board           |
| 18:57:28 26 J... | HTTP | → Request | GET    | http://localhost:3000/rest/languages                               |
| 18:57:28 26 J... | HTTP | → Request | GET    | http://localhost:3000/rest/admin/application-configuration         |

OWASP Juice Shop

Su Cesta (admin@juice-sh.op)

Precio Total: 0€

Tramitar pedido

(Obtendrás 0 puntos de bonificación por esta orden)

Filtros: Invertir Más filtros Todo FetchXHR Documento CSS JS Fuente img Medio Manifesto WS Wasm

| Nombre | Encabezados | Vista previa | Respuesta   | Iniciador | Tiempo | Cookies |
|--------|-------------|--------------|---|-----------|--------|---------|
| 1      |             |              | <pre>{   "status": "success",   "data": {     "id": 1,     "coupon": null,     "userId": 1,     "createdAt": "2025-06-26T21:35:40.563Z",     "updatedAt": "2025-06-26T21:35:40.563Z",     "status": "success"   } }</pre> |           |        |         |
| whoami |             |              | <pre>{   "status": "success",   "data": {     "id": 1,     "coupon": null,     "userId": 1,     "createdAt": "2025-06-26T21:35:40.563Z",     "updatedAt": "2025-06-26T21:35:40.563Z",     "status": "success"   } }</pre> |           |        |         |

Elementos Consola Fuentes Red Rendimiento Memoria **Aplicación** Privacidad y seguridad Lighthouse Grabadora

Aplicación
 

- Manifiesto
- Service workers
- Almacenamiento

Almacenamiento
 

- Almacenamiento local
  - http://localhost:3000
- Almacenamiento de la s...
- Almacenamiento de ext...
- Base de datos indexada
- Cookies
- Tokens de estado privado
- Grupos de interés
- Almacenamiento compa...
- Almacenamiento en cac...
- Buckets de almacenamie...

Servicios en segundo plano

http://localhost:3000
   
 Origen http://localhost:3000

| Key       | Value  |
|-----------|--|
| lsdtbxprt | "2025-06-26T21:45:49.973Z"   |
| token     | eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiaWF0Ij0iMjAyNS06-26T21:35:40.563Z"... |

## 4. Mitigación y Propuestas de Solución

Para cada vulnerabilidad identificada, se proponen las siguientes soluciones y medidas de seguridad, de acuerdo con las mejores prácticas de la industria y las recomendaciones de OWASP.

### A. Mitigación de Cross-Site Scripting (XSS):

- **Codificación de salida (Output Encoding):** Esta es la medida más efectiva. Antes de renderizar cualquier entrada del usuario en el HTML de la página, se deben codificar los caracteres especiales. Por ejemplo,

< debe convertirse en &lt; y > en &gt;. Esto asegura que el navegador interprete el código inyectado como texto simple en lugar de ejecutarlo.

### B. Mitigación de Inyección SQL:

- **Consultas Parametrizadas (Prepared Statements):** En lugar de concatenar la entrada del usuario directamente en las consultas SQL, se debe utilizar esta técnica. Esto garantiza que la base de datos trate los datos del usuario como valores literales, no como parte de la consulta, neutralizando así el ataque.

### C. Mitigación de CSRF:

- **Uso de Tokens CSRF:** Implementar un token único y secreto en cada formulario o solicitud HTTP que modifique datos. El servidor debe verificar que el token recibido coincida con el esperado para aprobar la solicitud.
- **Otras medidas:** Utilizar cookies SameSite y verificar el encabezado Referer en las solicitudes importantes.

## 5. Conclusión y Reflexión

Este ejercicio práctico me ha permitido aplicar mis conocimientos en pruebas de penetración en un entorno real. La experiencia de evadir filtros para lograr la explotación de XSS y de usar un proxy para inspeccionar el tráfico de la aplicación fue invaluable. Aprendí que la seguridad de una aplicación web depende de la implementación rigurosa de medidas de defensa en cada capa, desde la validación de entrada hasta la protección contra ataques específicos. La correcta documentación de los hallazgos y la propuesta de soluciones son tan importantes como la explotación misma, ya que guían a los desarrolladores a corregir las vulnerabilidades de manera efectiva. El ejercicio refuerza mi preparación para identificar, explotar y mitigar vulnerabilidades, lo cual es fundamental para una carrera en ciberseguridad.